PCI Express® Architectures for High Performance Compute and Storage Systems

Derek Percival

Senior Field Applications Engineer

Broadcom Ltd

Disclaimer

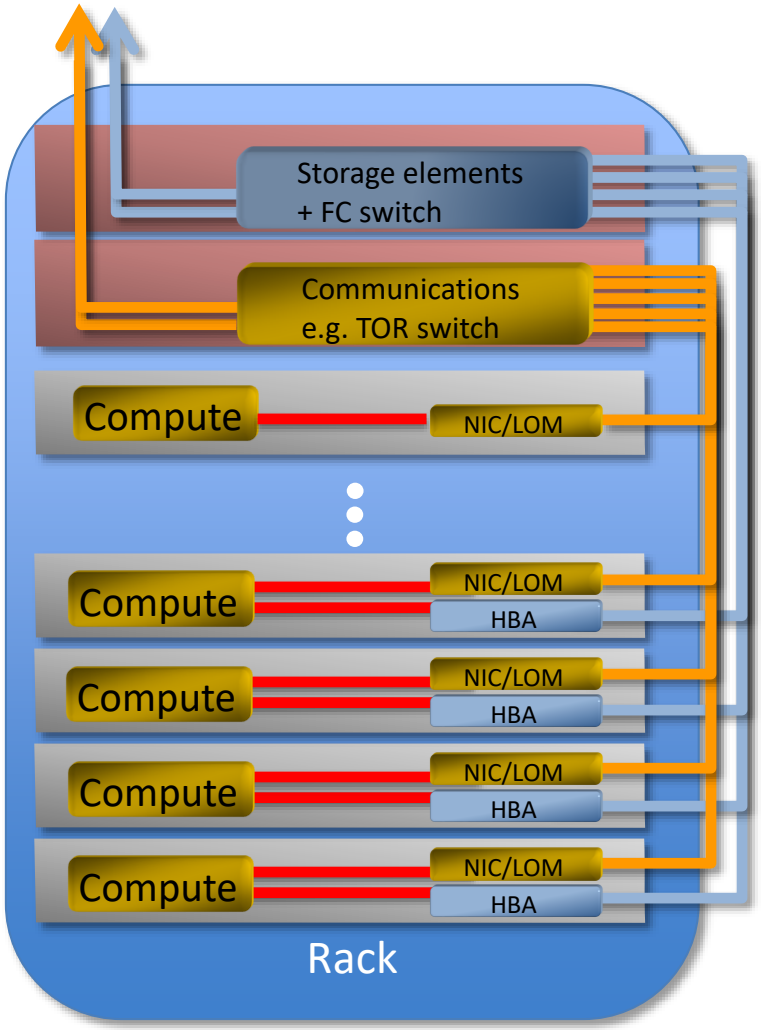
<u>Presentation Disclaimer</u>: All opinions, judgments, recommendations, etc. that are presented herein are the opinions of the presenter of the material and do not necessarily reflect the opinions of the PCI-SIG®.

Agenda

- **Current HPC and Storage architectures using PCIe® as the interconnect and their issues**
- **Ideal solution**
  - What we would like and what we can and can't do with older PCIe architectures
- **Advanced Architectures**
  - Virtual Switches and resource allocation issues
- **HPC using GPGPU**
- **Layered storage and resource pools**
- **Fabric switch based architectures**
- **Rack Scale architectures**
- **RSA and mixing fabrics**

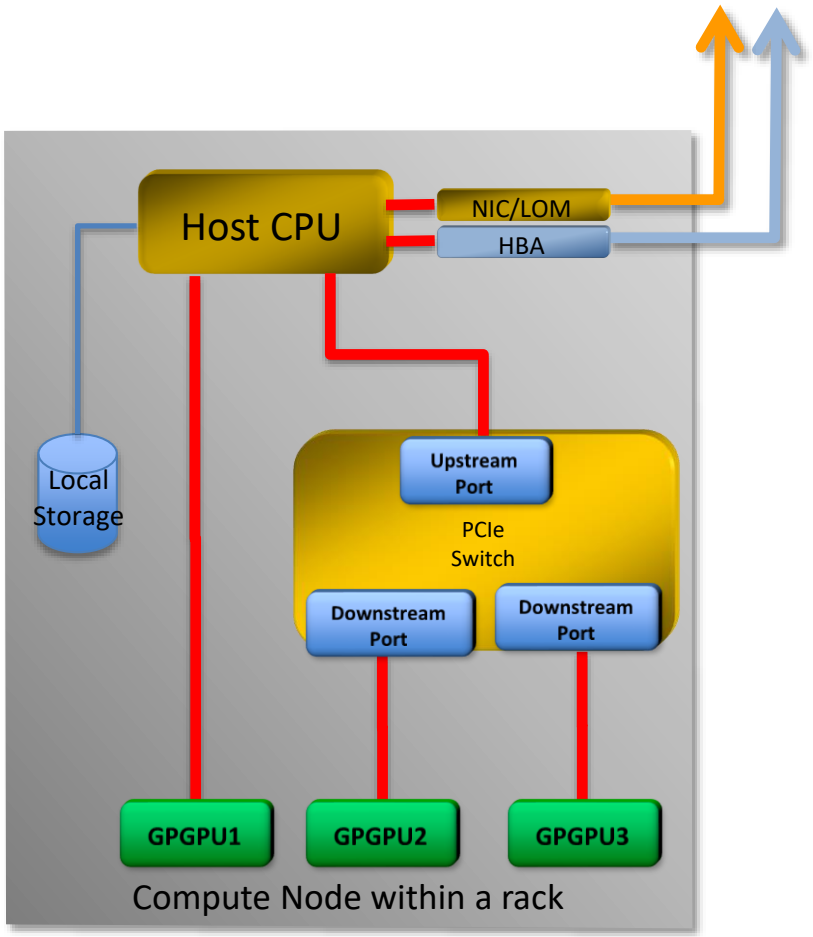
Current HPC Systems

- **Many current HPC designs use out of band interfaces**
  - Fibre Channel for Storage
  - Ethernet for communication
  - Adds latency and potentially bandwidth restrictions compared with PCIe
- **Multi-root complex systems must use out of band interfaces**
  - Change in communications type from memory mapped access to IP based communication
- **Difficult to share/change resources**
  - Requires additional software overhead
  - Changing from/to PCIe paradigm
- **Difficult to allocate resources to tasks**
  - Storage and compute nodes maybe in different racks adding latency



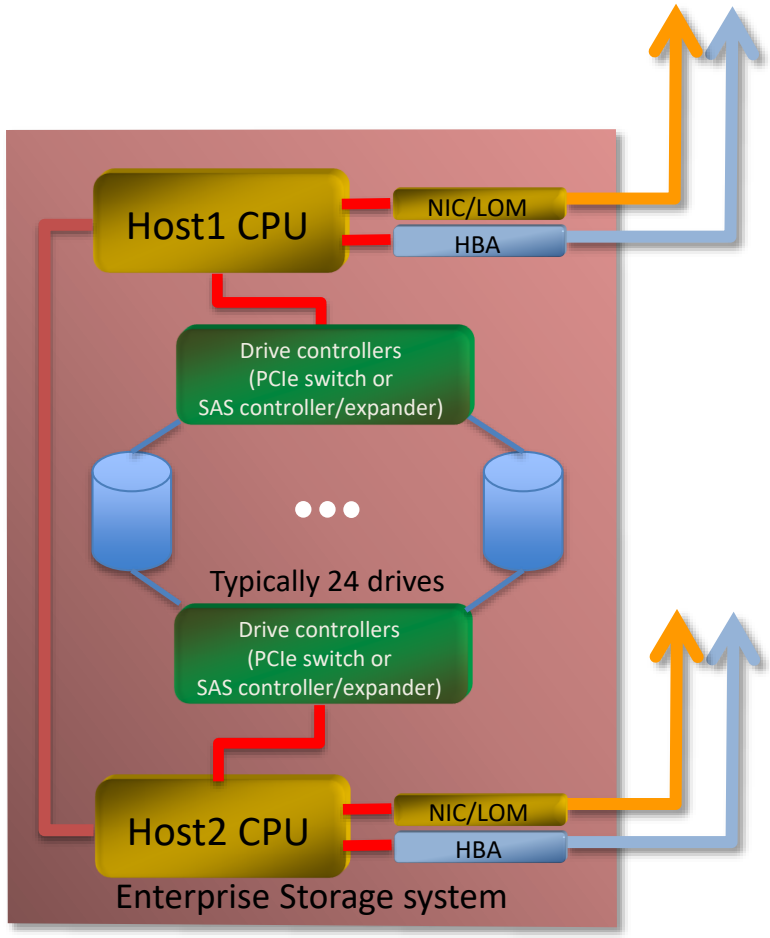
Current HPC Systems continued

- **GPGPU systems**
  - End point based General Purpose Graphics Processing Unit
    - Can share data and resources through the host CPU
    - Part of same PCIe tree
    - Peer to peer access between GPGPU's connected to the switch
      - Normally cannot do this if connected directly off root complex
  - Tough to share data for GPGPU's between root complexes
    - No direct connection in PCIe
  - Need to go out of band through Ethernet, Fibre Channel or some other non-PCIe bus to gain access to other, Hosts, GPGPU's and storage



Current Storage Systems

- **Current enterprise storage systems typically have drives associated with two hosts**
  - Dual fail over system
  - Multiport drives
  - Usually have data mirroring between the two hosts
  - Tough to share drives between multiple hosts
  - Out of band interfaces to compute nodes and other storage units



Current Storage Systems continued

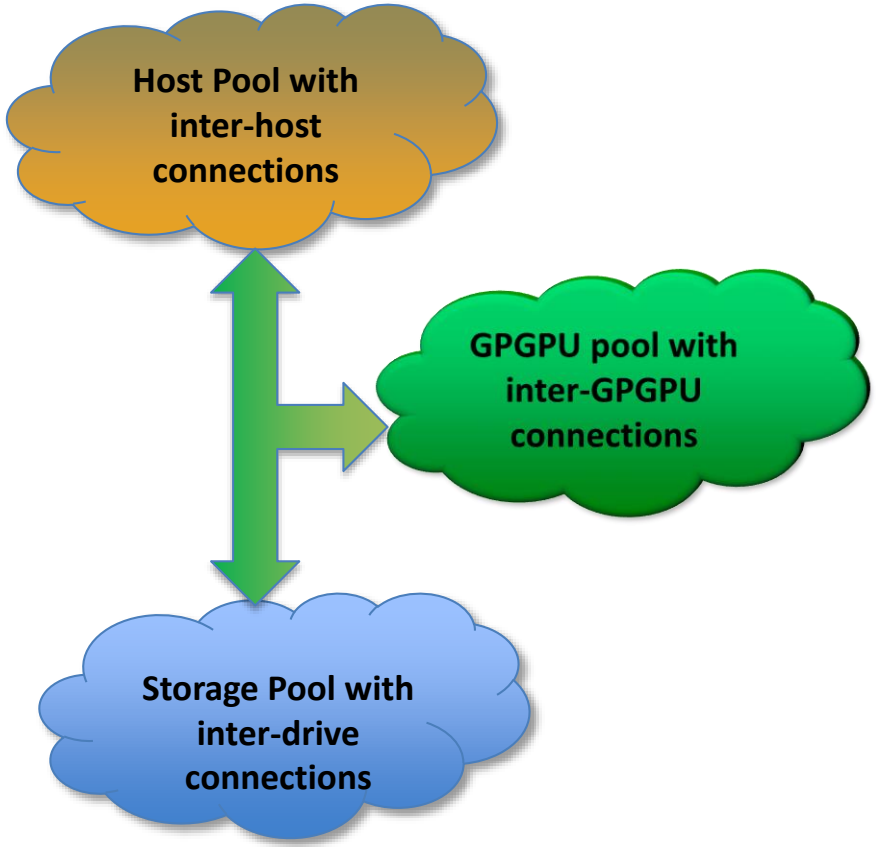
- **Details on points from previous foil**
  - Dual fail over system
    - Link between hosts may be side band (e.g. Ethernet) or PCIe using non-transparent interfaces
  - Multiport drives
    - Typically SAS from a SAS controller plus expanders
    - 2 port NVMe drives becoming available but expensive
      - Can create simple dual port drive from single port drives plus switches
  - Tough to share drives between multiple hosts
    - Requires moving data across a different fabric
    - Over fabric standard such as NVMe over Fabric allow remote pools but…
      - Still require additional latency to move between fabric types
      - Additional compute power required at target end to convert traffic to PCIe and route to appropriate drive

Current Storage Systems continued

- **New JBOF designs replacing JBOD**
  - Essentially a bunch of Flash drives connected to a switch
  - Access through PCIe cable interfaces or use an on board controller to connect to other fabric types
- **Cloud based services**
  - Starting to make more and more use of JBOF
  - Need to assign expensive high performance NVMe drives to hosts dependant on customer requirements
- **Warm storage with offload**
  - NVMe now starting to be used for warm storage but typically through other fabrics
- **Current architectures not flexible**
  - Direct attach means difficult to share or re-allocate drives

Ideal Solution

- **Multiple processing nodes**
  - Distributed across the system
  - Multiple connection paths
- **Multiple storage nodes**
  - Assignable to any processing node
  - Dynamically allocated
- **Any to any access**
  - Host to host
  - GPU to GPU
  - Host to any Storage etc.
- **Any topology**
  - Selected depending on the task
- **Resources allocated depending on task required**
  - Processing, GPU and Storage allocation
  - Dynamically reconfigurable
- **High bandwidth low latency paths between all elements**



What About Using PCIe

Positives

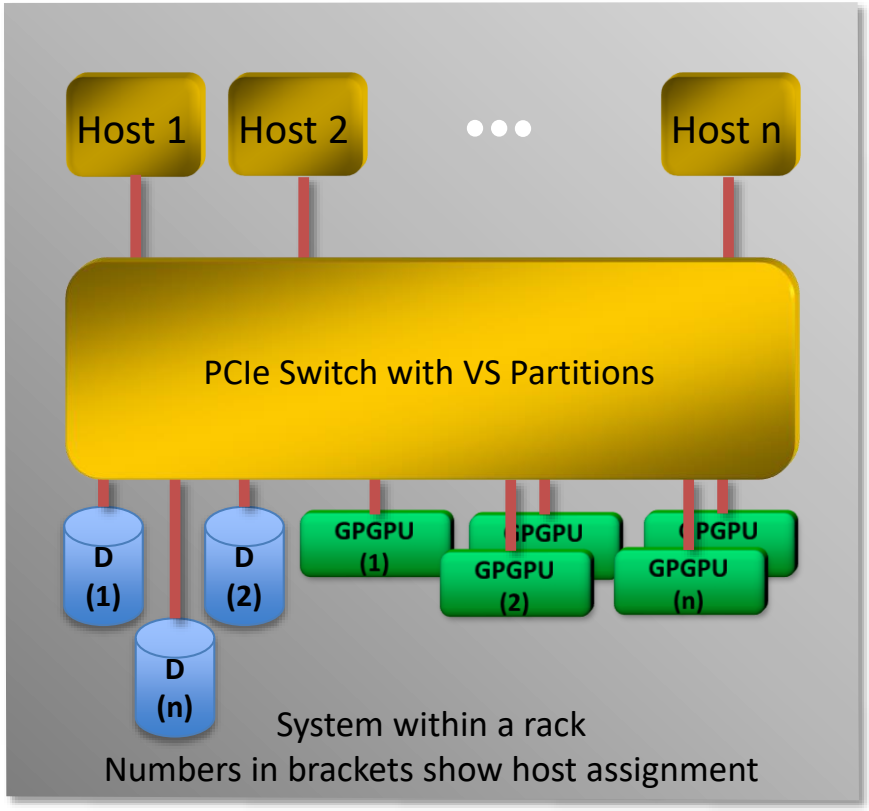
- **PCIe is already the primary high performance interface to host CPU's and GPU's**
- **PCIe provides a high bandwidth, low latency interface**
- **PCIe is well established**
- **Software drivers available for PCIe based devices**
- **Storage, networking and processing offload devices and cards already available with PCIe based systems**

Negatives

- **Doesn't scale well**
  - Limits in the number of buses and number of hosts which can be connected together
- **Host to host communication normally through other bus types**
  - Direct host to host communication through PCIe not allowed for downstream to downstream ports
- **Tough to re-configure the system**
  - Enumeration done at boot
  - Hot plug but tough to make large changes
- **Single host to adapter card topologies**
  - PCIe has always been an inverted tree topology
  - Other types of topology/fabric difficult to achieve

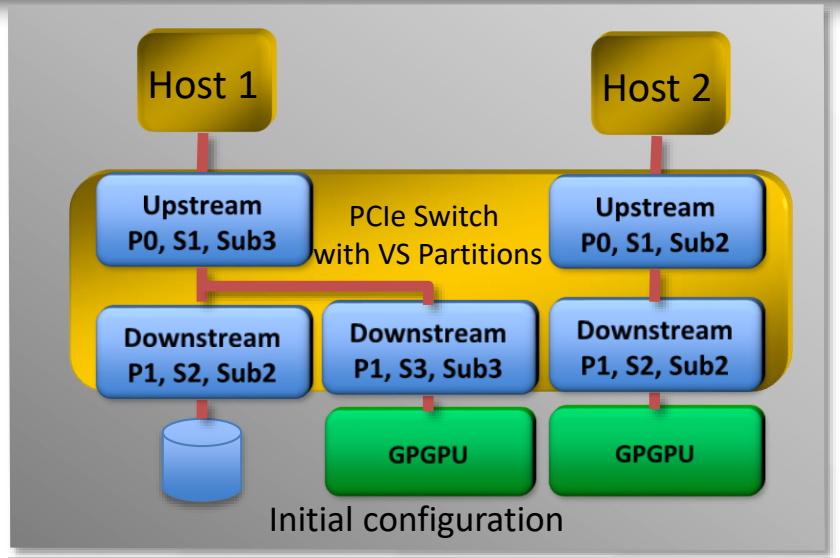
Advanced Architectures: PCIe and Virtual Switches

- **Multi host CPU based systems**
  - Switch partitioned
  - Each host gets resources within a Virtual Switch
  - VS effectively looks like multiple switches within one package
- **Storage pool available to multiple hosts**
  - Drives assigned depending on VS
  - Diagram shows one drive but can be many
- **Offload accelerators/GPGPU pool**
  - As per drives
- **Host to device allocation**
  - Typically this is determined before boot up e.g. using BMC
  - Normally a static attach
  - Difficult to dynamically configure
- **No PCIe host to host path except for low bandwidth message passing**

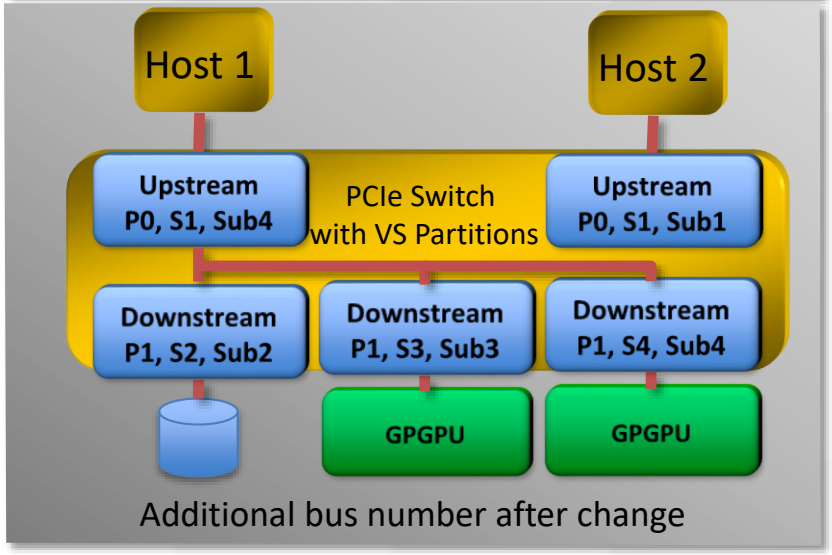


Difficulties of Resource Allocation

- **VS systems allow many static configurations**
- **Dynamic configurations are more difficult**
  - Effectively hot plug
  - Host OS needs to be hot plug aware
  - Resources need to be available
  - Address spaces and bus numbers may need to be increased/modified
  - Typically requires lots of OS work – see next slide
  - Although hardware changes quickly it's not fast to switch as OS requires time to accommodate changes



Initial configuration



Additional bus number after change

Overcoming Resource Allocation

- **Previous foil shows how on moving resources the OS needs to find space in the address map and bus numbers**
- **This may be difficult if the address range and/or bus number range adjoins to other resources within the system**
- **If there is no free address space or bus numbers then this will require**
  - All activity to the downstream devices to be stopped and in flight traffic drained
  - Unloading of the drivers
  - Re-enumeration of the subsystem
  - Re-loading of drivers
  - Activity can then be restarted
  - Potentially very slow to achieve and requires the OS to be capable of doing this

Overcoming Resource Allocation continued

- **Options to overcome this are**
  - Tell the OS to automatically provide gaps within the address and bus number ranges
    - OS needs to have that capability
      - May not have control of the OS used in the host
    - Okay in a well defined system but may be tricky if you don't know what OS will be used and what will be plugged in
    - Typically quite limited in the amount which can be added and the granularity of gaps within the address and bus ranges
  - Fixed enumeration
    - Often used in embedded systems
    - Requires the OS to have this capability or to re-work the kernel
    - Systems are well defined and controlled
    - Only devices with known resource requirements will be added

Overcoming Resource Allocation continued

- **Use the latest and next generation switches which allow pre-allocation of resources using synthetic end points**
  - This latest system still has some limits but allows any OS to be used
  - No need for the OS to be capable of adding gaps in enumeration or to use fixed enumeration
  - Size, and number of end points is fully configurable and usually set up in software
  - Requires a management controller to be running before the hosts boot

Limitations of Virtual Switches

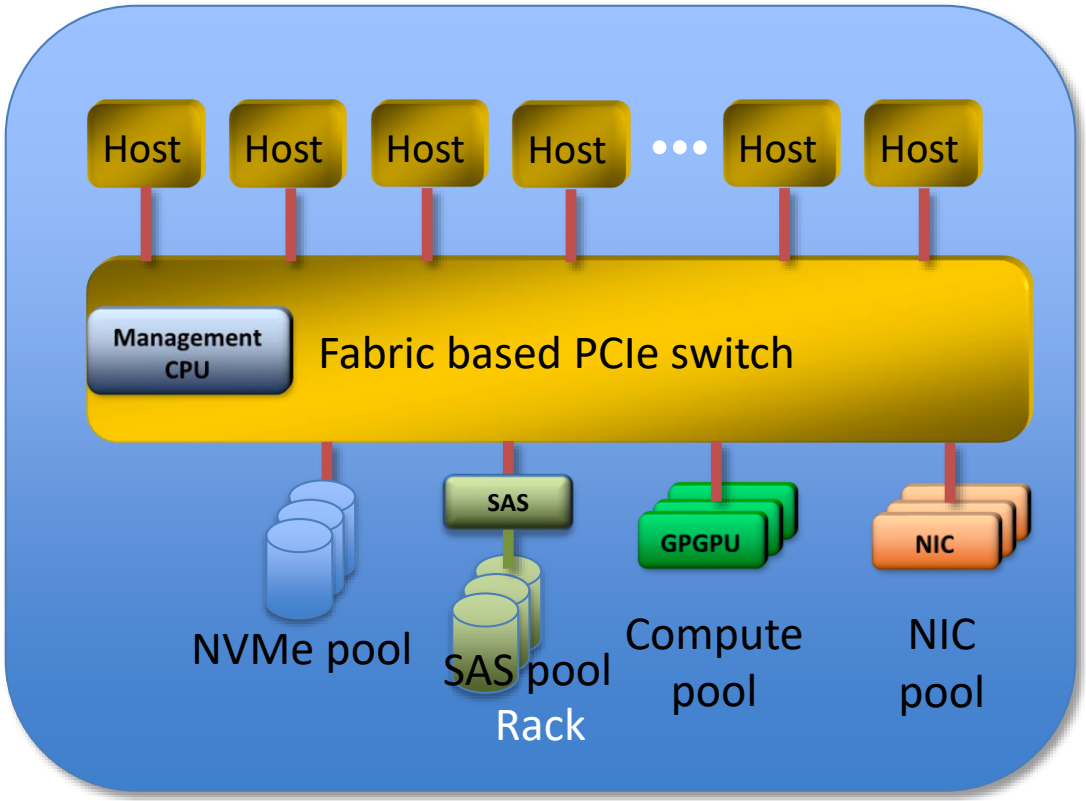
- **Even overcoming resource allocation there are still limitations to the VS approach**
- **Compute resources such as GPGPU are restricted to one VS**
  - Moving data from one GPU to another on the same switch but different VS requires use of a different fabric
    - Typically Ethernet which means the host needs to get involved
  - GPGPU's cannot be shared between hosts
- **Same issue for storage pools**
  - High performance NVMe drives may only be partially used
- **Second or third interconnect structure required**
- **Virtual Switches are a good step but only solve part of the problem**

HPC Using GPGPU and Resource Sharing

- **Latest generation fabric based switches allow much more flexibility**
- **Multiple GPGPU nodes may pass data between each other**
- **SR-IOV or Multi-function devices can shared between hosts**
- **Multiple storage networks can be created**
  - SR-IOV based SAS controllers can be shared between hosts
  - SR-IOV based NVMe drives can be shared between hosts
- **Synthetic end points can be created**
  - Allowing pre-allocation of resources
  - Dummy drivers to allow control of systems
- **Access between hosts**
  - Allows controlled data passing as well as synchronisation for shared tasks
- **Current systems build per task**
  - New systems may be able to dynamically allocate resources during tasks
- **Still limitations in scalability of shared resources due to bus number limitations**
  - Will be solved in next generation of devices

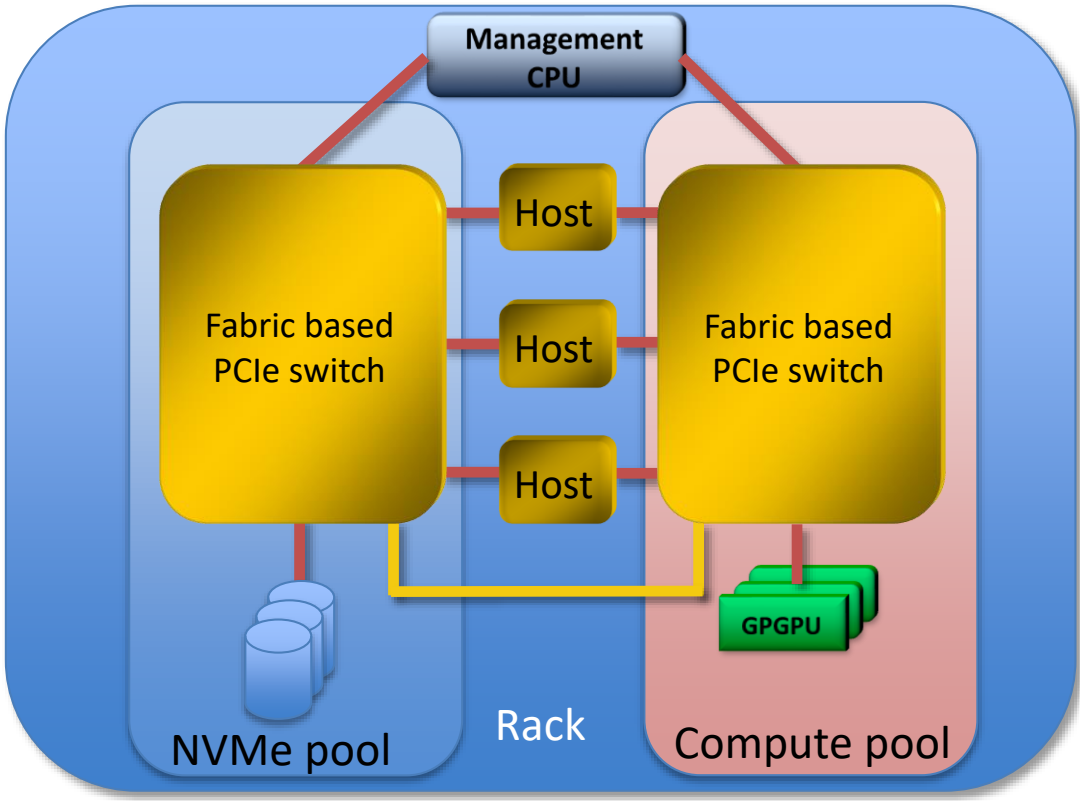
HPC Using GPGPU and Resource Sharing

- **Multiple resource pools shared between hosts within a rack**
- **SR-IOV functions or functions from multifunction devices associated with different hosts**
- **Synthetic end points ensure resources available in OS to accommodate different compute and storage tasks**
- **All interfaces use PCIe so intercommunication at full Gen x speed with low latency**
- **Allocation of resource controlled by management CPU**

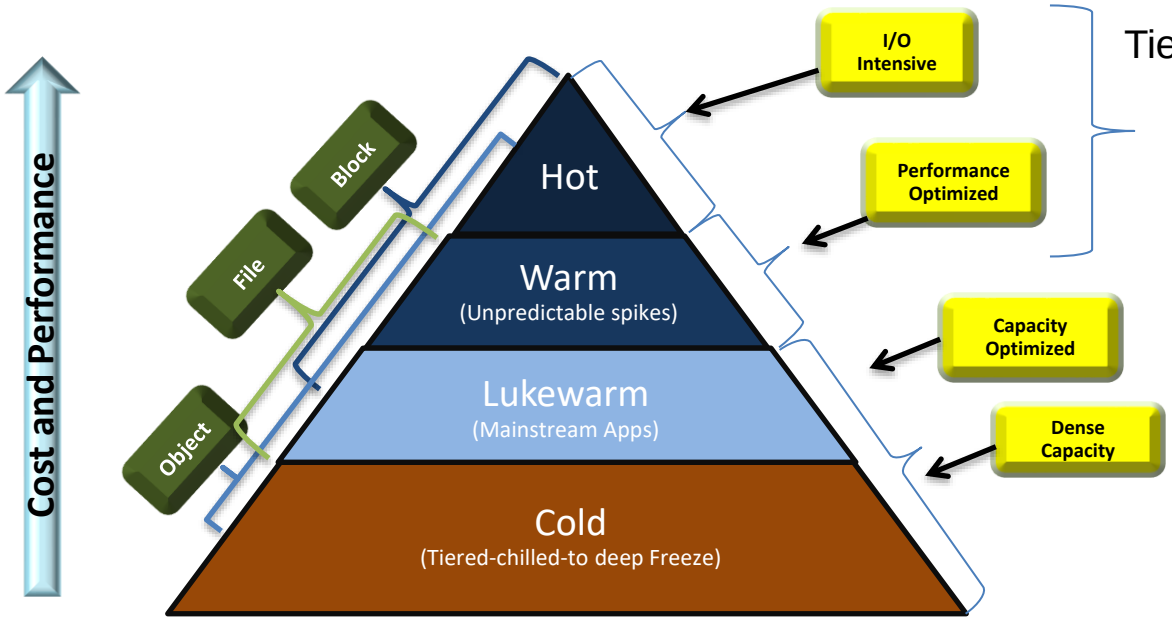


Separation of Storage and Compute Blocks

- **More common architecture has storage and compute in separate chassis**
- **Same could be applied to any pool**
- **Can still assign any resource to any host**
- **Devices within a pool can communicate with each other**
- **Communication of devices between pools not possible unless create additional links**
  - This would normally be a crosslink but in fabric switches uses a special fabric port to allow such a topology to be created



Layered Storage and Resources



Tier 0-1: NVMe Hot Caching

- I/O and performance intensive applications
- Low latency, High data BW for block store

Tier 3: Cold Spinning HDD

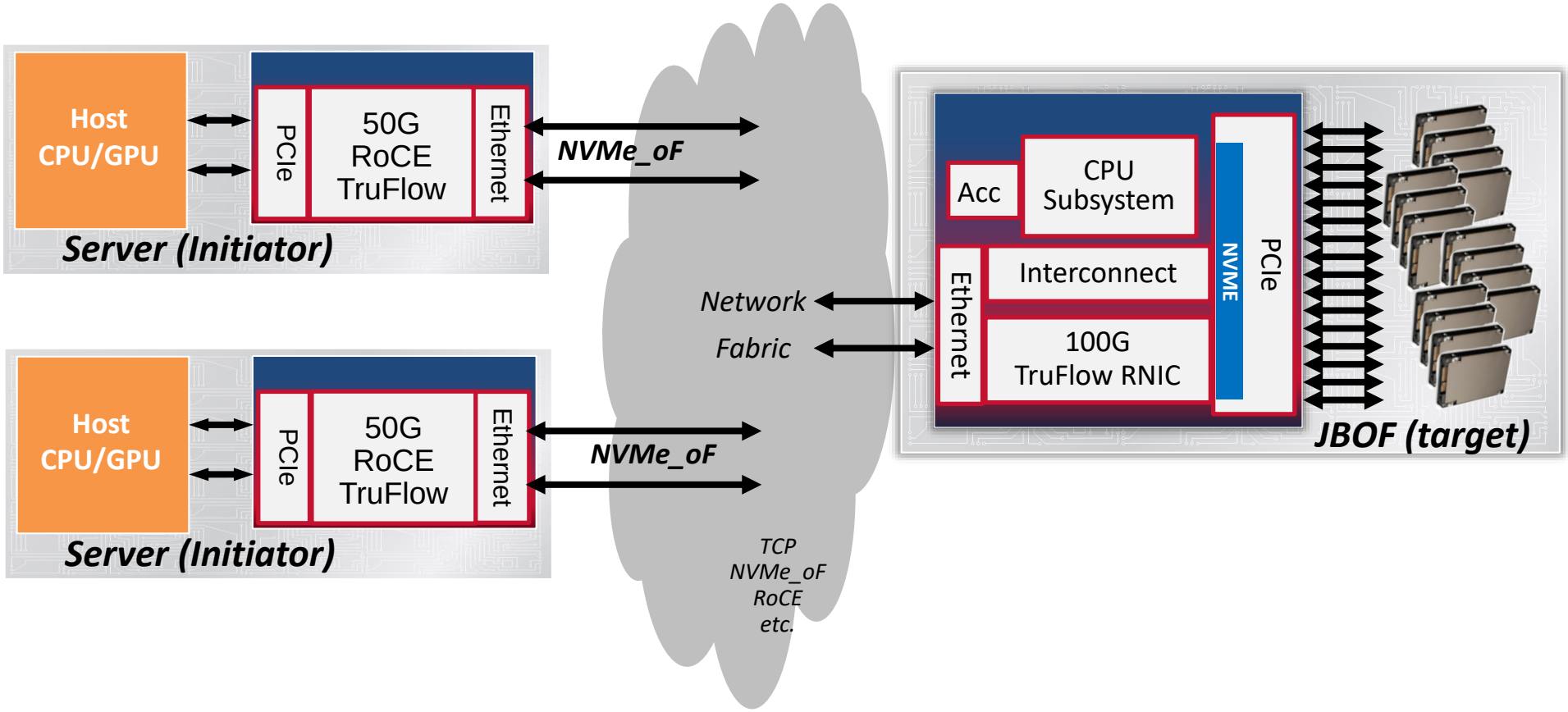
- High density active tier can require high I/O BW, can be SSD based

PCIe in Layered Storage and Resources

- **All tiers of storage can be PCIe based**
  - SAS cards generally have a PCIe interface
  - New systems are using NVMe drives for cold and warm storage
    - NVMe drives used for fast access then data moved to cold storage units
    - Effectively a cache type approach but for much larger quantities of data
  - May need to move beyond the rack for greater storage and sharing with different racks or different buildings
    - Need to move PCIe based solutions over Ethernet
      - Effectively results in converging fabrics of PCIe and Ethernet for storage
      - Low latency within the rack higher latency when moving between racks

Storage over Converged Ethernet

- **SSD disaggregation with NVM_oF**
- **RoCE and NVM_oF in virtualized network**



Resource Pools in a Converged System

- **Resource pools of**
  - Host processors
  - GPGPU
  - NVMe
  - Slower Storage e.g. SAS
  - NIC's
- **Ideal architecture allows all to access anything simultaneously**
  - Difficult to achieve as there would be potential contention between hosts accessing the same end point
  - Can be achieved using shareable devices

Types of Pool Assignment

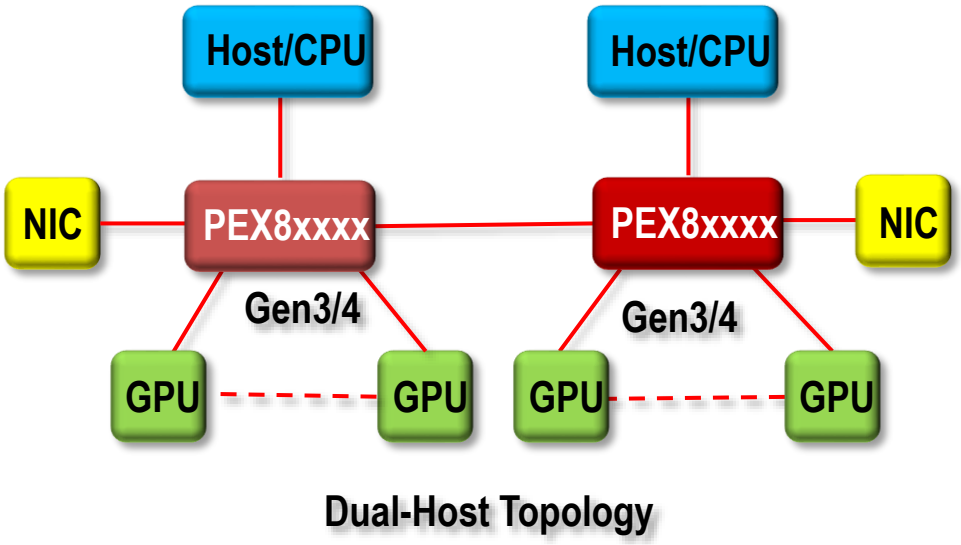
- **There are many types of pools and within these pools there can be different types of assignment. Some examples are**
  - Full Drive/Device
    - Entire device or end point is owned by one host
  - Partial Drive/Device
    - One or more host shares the end point
  - High Availability
    - Multiple versions of the end point or data perhaps through mirroring or multi-cast
    - Multi-port devices may also be used to accommodate this

Using Fabric Based Switches

- **Fabric based PCIe switches allow all types of pool assignment**
  - Initial configuration by management processor determines available resources
  - Host enumeration discovers only those resources assigned to that host
    - This may include any of the previously mentioned types ouf resource
    - Only limitation is the type of end points available
    - Synthesised tree of available resources provided to each host
    - Dummy trees and end points can be created to allow alternate pools of resources and switching of resources between hosts

Example: GPU Server Redundancy

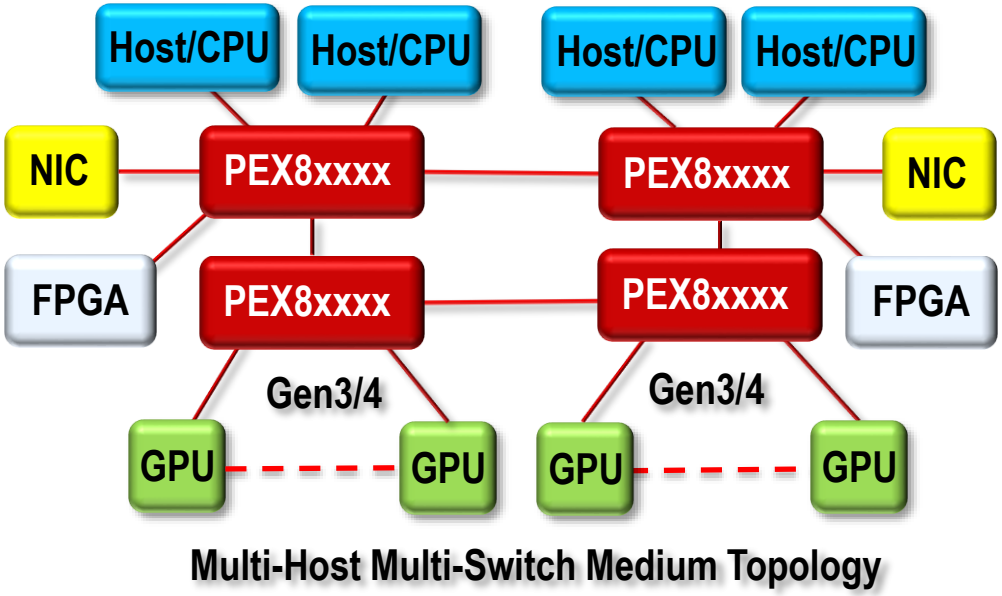
- **Redundancy:**
  - Dual host for fail-over
  - Active-active or Active-passive host
  - Extend ports through fabric connection – eliminate hierarchy
    - Host can be assigned resources from either switch
    - End points can communicate between switches
  - Dynamic allocation of GPUs to hosts
  - Failover using standard hot add/remove



Dual-Host Topology

Hyper-Scale Server with Resource Pooling

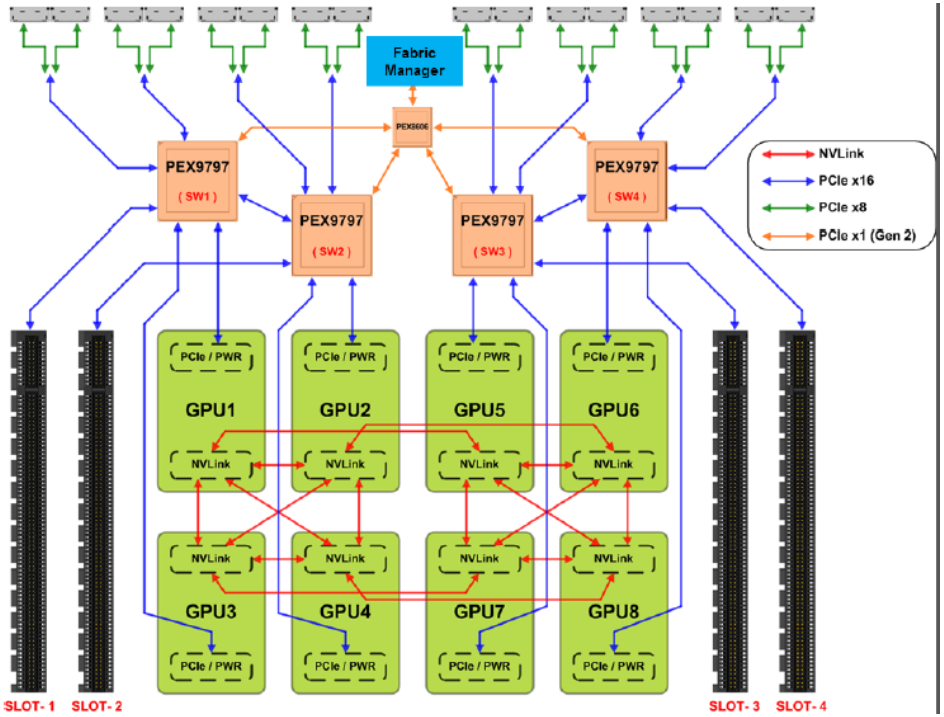
- **Resource Pooling:**
  - Up to 4 hosts, 8 GPU, NICs & FPGA
  - Four 96-lane PCIe Switches
  - Dynamic allocation of resources to any host or FPGA
  - Dynamic allocation of devices to hosts
  - Move data between devices with on-chip DMA
  - Multiple paths through the fabric for host to EP and EP to EP



Multi-Host Multi-Switch Medium Topology

Microsoft Project Olympus

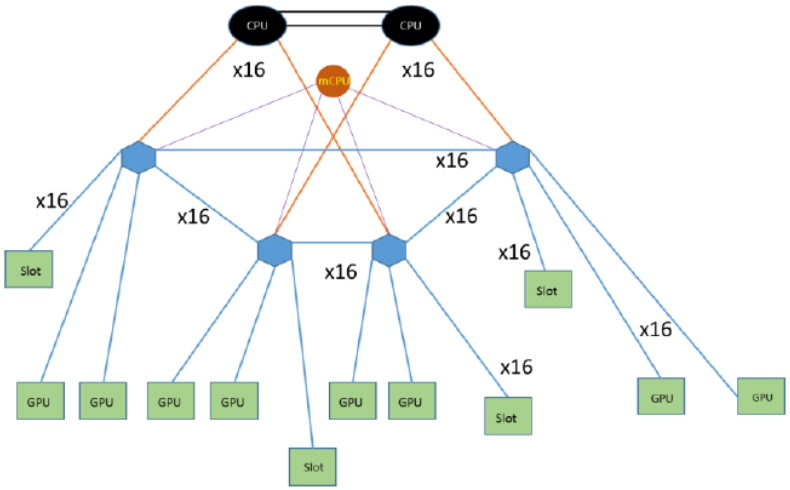
- **Hyperscale GPU Accelerator Chassis**
  - Flexible PCIe Interconect
  - GPGPU to Host Via high BW PCIe links
  - Peer to peer through
    - PCIe links
    - NV Link
  - Rack scale out through IB links
  - Configurable CPU <-> GPU ratios depending on application E.g.
    - 2 CPU to 8-16 GPU for deep learning
    - 4-8 CPU to 8 GPU for HPC
    - Performance is optimised depending on workload



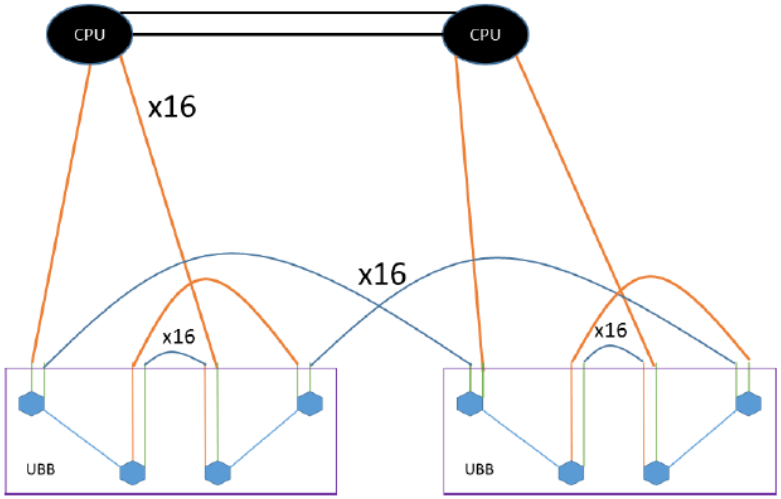
Source: Microsoft Project Olympus Hyperscale GPU Accelerator – Presented to OCP U.S Summit 2017

Microsoft Project Olympus continued

- **High Performance peer to peer bandwidth**



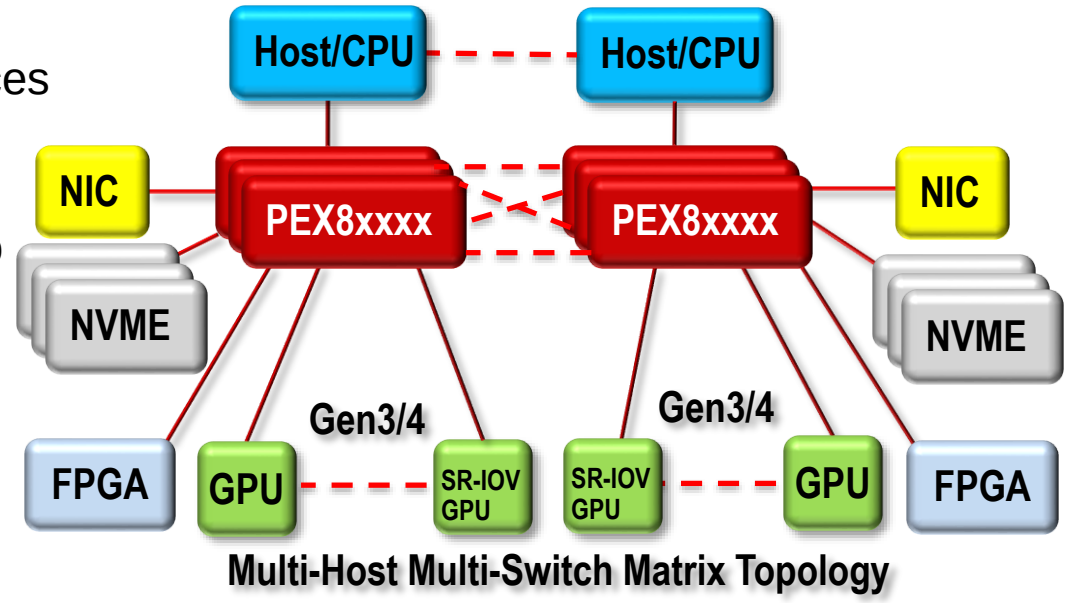
- **Chassis to chassis interconnect**



Source: Microsoft Project Olympus Hyperscale GPU Accelerator – Presented to OCP U.S Summit 2017

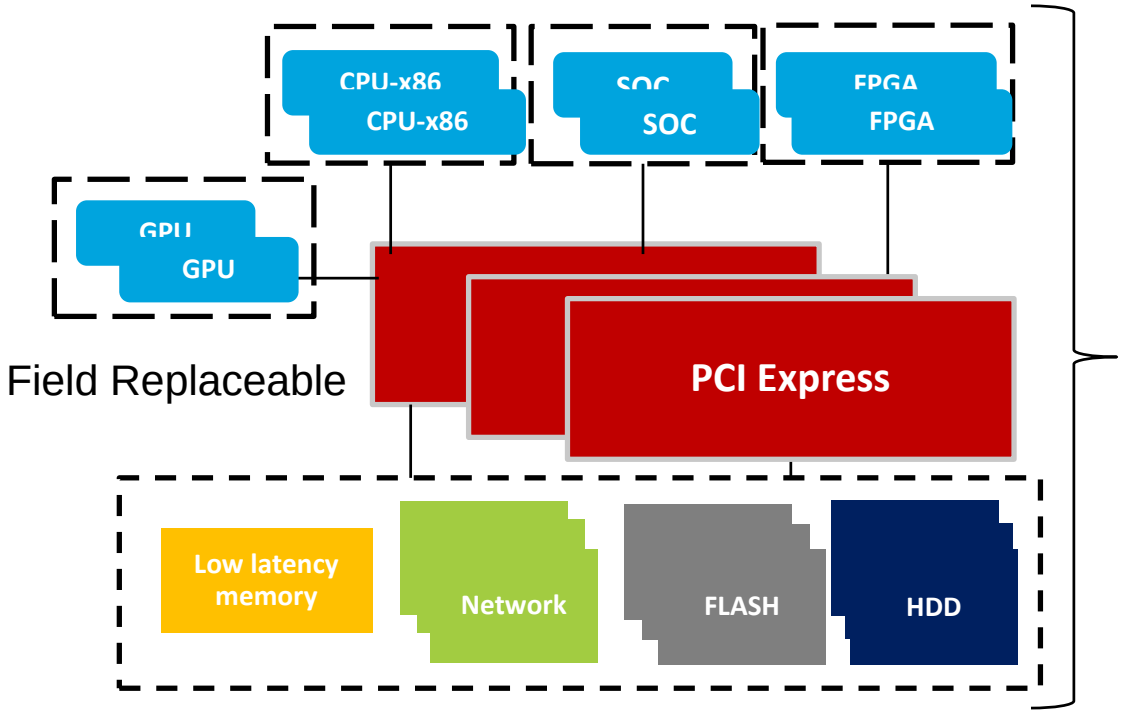
Scalable/Composable System Using PCIe Fabric

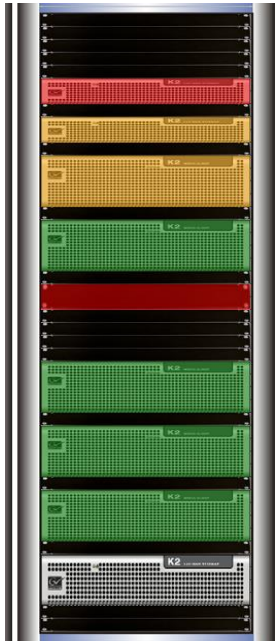
- **Resource Pooling & Sharing**
  - Up to 72 Hosts, GPUs, NICs & FPGA
  - Dynamic allocation of resources to any Host or FPGA
  - Allocate any-to-any
  - Move data from any device to any device with on-chip DMA
  - Share multi-function GPU
  - Share array of NVMe devices
  - Share multi-function NVMe devices
  - Mesh fabric topology for bandwidth and fail over



Rack Scale Architecture
Composable Data Centre Rack

- **Enable PCIe Connected Bare-metal Resource Pool for Hyperscale System**
- **Accelerate HPC, AI, Data Analytics Apps with low latency rack-level data transfers**





Summary for HPC in Rack Scale Architectures Using PCIe

- **Storage, HPC and GPGPU systems can benefit from more flexible architectures**
- **New PCIe based switches offer many benefits including**
  - High bandwidth/low latency paths
  - Scalability
  - Dynamic re-configuration/provisioning of resource pools
  - Many fabric topologies within the PCIe paradigm
  - Larger systems far beyond the normal 256 bus limit
- **PCIe fabrics can be scaled to even greater sizes by mixing other fabric types**
  - Top of rack switches to enable 100GE, Fibre Channel and InfiniBand access to other racks or rows
- **Devices and systems available today from multiple suppliers**
  - E.g. Inventec and Liqid offer turn-key composable systems based on PCIe
  - Future devices and systems will offer even greater flexibility and scalability

Thank you for attending the PCI-SIG Developers Conference Israel 2017.

For more information please go to www.pcisig.com